(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 100 47 338 A 1**

(51) Int. Cl.[7]: G 06 F 17/21

(21) Aktenzeichen: 100 47 338.5
(22) Anmeldetag: 25. 9. 2000
(43) Offenlegungstag: 18. 4. 2002

DE 100 47 338 A 1

(71) Anmelder:
Siemens AG, 80333 München, DE

(72) Erfinder:
Heuer, Joerg, 81539 München, DE; Hutter, Andreas, 81539 München, DE; Niedermeier, Ulrich, 80796 München, DE

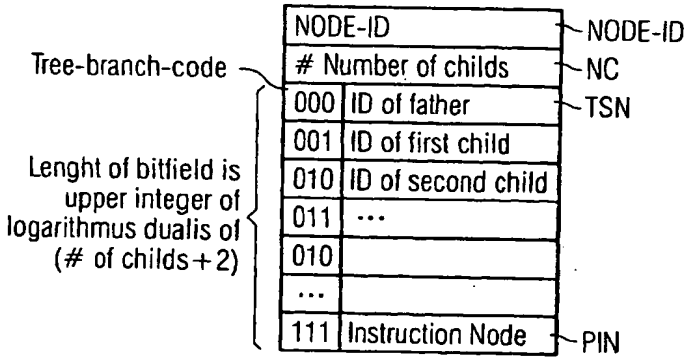
(56) Entgegenhaltungen:
http://www9.org/w9cdrom/154/154.html (19.09.2000), Millau: an encoding format for efficient representation and exchange of XML over the Web", <Marc Girardot and Neel Sundaresau:>;

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zur Datenkompression von strukturierten Dokumenten und Anordnung zur Durchführung des Verfahrens

(57) Der Anmeldungsgegenstand betrifft ein Verfahren bzw. eine Anordnung zur Datenkomprimierung von XML-basierten Dokumenten, insbesondere MPEG-7-Dokumenten, bei dem Beschreibungsschemata, Instanzen von Dokumenten und Änderungen dieser Instanzen z. B. unter Zugrundelegung eines Modells eines Zustandsautomaten in Anweisungen an diesen Zustandsautomaten zur Bildung dieser Strukturen übergeführt werden, wobei diese Anweisungen im Wesentlichen nur relative Adressen aufweisen und dadurch eine Komprimierung erreicht wird.



DE 100 47 338 A 1

Beschreibung

[0001] Die Erfindung betrifft ein Datenkompressionsverfahren von XML-basierten Strukturierten Dokumenten, wie sie beispielsweise bei einem Entwurf für MPEG-7 vorgesehen sind. Die XML (Extensible Markup Language) ist ein Standard für die Definition von Dokumentenstrukturen und dient zur Darstellung strukturierter Daten in einem Textfile und bildet beispielsweise die Basis für die Sprache XHTML. MPEG-7 ist ein Entwurf zu einem ISO/IEC-Standard für eine Beschreibungssprache von multimedialen Inhalten. Diese XML-basierten strukturierten Dokumente basieren auf einer Menge von Strukturierten Elementen – im folgenden auch "Schema" genannt – wie sie zum Beispiel mit Hilfe von Document Type Definition (DTD), XML Schema oder Multimedia Description Schemes (DS) spezifiziert werden können.

[0002] Diese strukturierten Dokumente liegen typischerweise als Textdatei vor, was einen erhöhten Speicherbedarf und für eine Übermittlung einen erhöhten Bandbreitebedarf bedeutet. Eine Möglichkeit, dieses Speicherplatz- oder Bandbreitenproblem zu lösen, ist eine Kompression der Daten. Traditionelle Datenkompressionsverfahren erreichen gute Kompressionsraten bei großen Textfiles, sind aber weniger effektiv bei kleineren Files, wie sie beispielsweise bei E-Business-Anwendungen im Internet vorkommen. Darüber hinaus können diese Kompressionsverfahren auf einen kontinuierlichen Datenstrom zurückgreifen. Um wirklich effizient arbeiten zu können, benötigen sie jedoch beispielsweise das gesamte Datenfile. Dies kollidiert jedoch mit den Realzeitbedingungen eines Netzes und ermöglicht keinen Wahlfreien Zugriff auf die codierten Daten.

[0003] In der Internetveröffentlichung http:/ /www9.org/w9cdrom/154/154.html vom 19.09.2000 ist auf den Seiten 1 bis 24 ein Codierungsformat für eine effiziente Repräsentation und einen Austausch von XML über das Netz unter dem Namen "Millau" veröffentlicht. Das Millau-Format wurde dazu entwickelt, XML-Dokumente in einer kompakten Art und Weise zu repräsentieren. Dies geschieht dadurch, dass "tokens" anstelle von "tags" der XML-Sprache und Attribute anstelle von Strings der XML-Sprache verwendet werden.

[0004] Die der Erfindung zugrunde liegende Aufgabe besteht nun darin, ein Verfahren zur Datenkompression von Strukturierten Dokumenten anzugeben, bei dem eine möglichst gute Kompression bei gleichzeitig hoher Flexibilität hinsichtlich der Datenstruktur erreichbar ist.

[0005] Dieses Problem wird hinsichtlich des Verfahrens erfindungsgemäß durch die Merkmale des Patentanspruchs 1 sowie durch die Merkmale des Patentanspruchs 4 und hinsichtlich der Anordnung erfindungsgemäß durch die Ansprüche 13 und 14 gelöst.

[0006] Aus den weiteren Ansprüchen ergeben sich bevorzugte Ausgestaltungen der Erfindung.

[0007] Im folgenden werden die hier verwendeten Ausdrücke "Strukturierte Elemente" und "strukturiertes Dokument" bezüglich Ihrer Eigenschaft definiert:

- Strukturierte Elemente können eine Menge an zulässigen Verknüpfungen mit weiteren Strukturierten Elemente spezifizieren. Des Weiteren können Strukturierte Elemente weitere zulässige Inhalte spezifizieren.
- Ein strukturiertes Dokument enthält Instanziierungen der strukturierten Elemente bestehend aus einer Auswahl an Verknüpfungen weiteren Inhalten.

[0008] Die Erfindung basiert im Wesentlichen darauf, dass Strukturierte Elemente als Zustandsmuster (Knoten) eines Zustandsautomaten interpretiert werden können. Die Verknüpfungen einer Menge an Strukturierten Elementen können als Zustandsübergänge des Zustandsautomaten aufgefasst werden. Demnach kann die Struktur eines strukturierten Dokumentes als Liste von Zustandsübergängen interpretiert werden. Die Struktur eines MPEG-7 oder eines anderen XML-basierten Dokuments kann dementsprechend als Liste von Zustandsübergängen repräsentiert und binär effizient codiert werden. Der Vorteil dieses Modells ist, dass implizite Syntaxbeschränkungen sehr effizient ausgenutzt werden können, da in jedem Knoten nur die möglichen Folgezustände codiert werden und dadurch nur sehr kurze Codes für die Baumstruktur des Dokuments erforderlich sind. Des weiteren ist der Codierprozess des Modells sehr flexibel und erweiterbar, da neue Eigenschaften und Anforderungen einfach durch die Definition neuer Zustands-Prototypen oder Befehle für einen Befehlsknoten integriert werden können. Zudem basieren Definitionen von Mengen von Strukturierten Elementen ebenfalls auf strukturierten Dokumenten, so dass mit der hier vorgestellten Codierung diese ebenfalls binär codiert werden können, zum Beispiel stellt ein XML Schema Dokument ebenfalls ein XML Dokument dar.

[0009] Durch die Erfindung werden beispielsweise folgende Vorteile erreicht:

1. Eine effiziente binäre Darstellung von strukturierten Dokumenten und/oder Strukturierten Elementen.
2. Eine kompakte Codierung der Struktur eines Dokuments.
3. Eine Unterstützung der dynamischen Veränderung des strukturierten Dokuments und/oder Strukturierten Elements.
4. Eine Unterstützung für den wahlfreien Zugriff auf ein codiertes Dokument.
5. Eine Unterstützung für die progressive Übertragung, d. h. die Reihenfolge der Elemente eines Dokuments im Datenstrom ist variabel.
6. Eine Unterstützung einer skalierbaren Binärrepräsentierung eines Dokumentes, d. h. es wird Basisinformation übertragen, die die grundlegenderen Elemente beinhaltet und eine oder mehrere Datenströme mit zusätzlichen Details.
7. Eine Trennung der Struktur des Dokuments vom Inhalt wird ebenso unterstützt, wodurch der Inhalt mit einer separaten darauf spezialisierten Methode komprimiert werden kann.

[0010] Nachfolgend wird anhand der Zeichnung ein Ausführungsbeispiel der Erfindung näher erläutert. Dabei zeigt:

[0011] **Fig.** 1 einen Baumstrukturknoten eines Zustandsautomaten,

[0012] **Fig.** 2 einen Instruktionsknoten eines Zustandsautomaten

[0013] **Fig.** 3 einen Attributknoten eines Zustandsautomaten

[0014] **Fig.** 4 ein XML-Schema für ein Beispiel einer Menge von Strukturierten Elementen,

[0015] **Fig.** 5 ein zum Beispiel von **Fig.** 4 gehöriges binäres Format der Struktur,

[0016] **Fig.** 6 Anweisungen in Form eines Bitcodes zum Aufbau der Struktur von **Fig.** 4 bzw. 5 mit entsprechenden Kommentaren,

[0017] **Fig.** 7 ein Beispiel eines Strukturierten Dokuments in Form eines XML-Dokuments,

[0018] **Fig.** 8 Anweisungen in Form eines Bitcodes zur Erzeugung des in **Fig.** 7 beschriebenen Strukturierten Dokuments,

[0019] **Fig.** 9 eine binäre Darstellung des Strukturierten

Dokuments von **Fig.** 7 und

[0020] **Fig.** 10 Anweisungen in Form eines kommentierten Bitcodes zur Änderung eines Strukturierten Dokuments.

[0021] In den **Fig.** 1 bis 3 sind Strukturierte Elemente (Knoten) zum Aufbau einer hier baumartigen Struktur für ein strukturiertes Dokument gezeigt. Die Struktur ist in den meisten Fällen aus mehreren dieser Strukturierten Elementen gebildet, kann aber auch aus nur einem dieser Strukturierten Elemente gebildet sein.

[0022] In **Fig.** 1 ist ein Baumstrukturknoten TSN dargestellt, aufgebaut aus einer lokalen Identifikationsnummer NODE-ID, einen Speicherplatz für die Anzahl NC von Kindknoten, eine Zelle mit der Adresse 000 für einen Zeiger auf einen Vaterknoten und Zellen mit den Adressen 001 bis 110 für Zeiger auf Kindknoten sowie eine Zelle mit der Adresse 111 mit einem Zeiger auf einen Befehlsknoten PIN. Die Knotenbezeichnung NODE-ID definiert den Knotentyp in eindeutiger Weise. Der Knotentyp kann entweder ein vordefinierter Knotentyp sein, der beispielsweise aus Standardelementen eines standardisierten MPEG-7-Beschreibungsschemas definiert wurde oder aber ein abgeleiteter oder erweiterter Knoten mit einer lokalen Identifikationsnummer. Aus der Anzahl der Kindknoten wird die Länge des Bitfeldes berechnet, das zur Adressierung der Kindknoten benutzt wird. In diesem Beispiel sind mit einer 3-Bitadresse bis zu 6 Kindknoten möglich. Mit der Adresse alle Bits gleich Null, hier 000, erfolgt ein Übergang in eine höhere Ebene einer Dokumentenhierarchie. Das letzte Codewort mit allen Bits gleich Eins, hier 111, hat die spezielle Bedeutung, die den Zustandsautomat in den Befehlsknoten zu bringen. Das Wesentliche ist dabei, dass bei der Übermittlung bzw. der Speicherung des Dokuments nicht die Knotenidentifizierungen der Kindknoten übertragen werden, sondern nur Anweisungen in Form eines Baumverzweigungs-Codes TBC, wodurch die Codeworte sehr kurz und ihre Länge unabhängig von der Tiefe des Knotens innerhalb des Baumes ist.

[0023] In **Fig.** 2 ist ein Befehlsknoten IN dargestellt, der hier abhängig von einer 4-Bitadresse eine Escape-Funktion, eine Initialisierung eines neuen Dokuments und beispielsweise weitere Befehle zum Hinzufügen und Löschen von Knoten und Gruppen sowie weitere Funktionen enthält. In einem Instruction Set werden verschiedene Verfahren definiert, mit denen ein Strukturiertes Dokument durch Erweiterung oder Beschränkung verändert werden kann. Um diese Funktionalität zu unterstützen, wird der Typ eines Befehlsknotens initiiert. Wenn ein Knoten in einem Strukturierten Dokument verändert werden soll, werden die entsprechenden Codeworte gesendet, um in dem instanziierten Dokumentbaum an den betreffenden, zu ändernden Knoten zu gelangen. Dies kann entweder ausgehend von der Dokumentwurzel erfolgen, also durch einen absoluten Pfad, oder mit Hilfe eines Verweises in einer Zeigertabelle oder mit einem relativen Pfad ausgehend von der aktuellen Position im Baum. Dann wird das Codewort für den Befehlsknoten IN gesendet, d. h. der letzte Eintrag im Baumverzweigungsknoten, in diesem Beispiel also 111. Das folgende Codewort definiert die erste auszuführende Instruktion sowie gegebenenfalls die zugehörigen Parameter. Auch weitere Instruktionen können folgen. Wenn die Escape-Instruktion, hier also 000, gesendet wird, schaltet der zugrundegelegte Zustandsautomat zurück in den Vaterknoten, von dem aus der Befehl aufgerufen wurde. Der Befehls- oder Instruktionstyp legt fest, wie viele Parameter, hier P1 und P2, gelesen werden sowie deren Format und Semantik. Die Parameter definieren z. B. die Position, an der ein Knoten eingefügt oder gelöscht werden soll sowie die Knotenidentifizierung bei neu hinzugefügten Knoten. Neben den in **Fig.** 2 dargestellten Basisinstruktionen sind beispielsweise noch folgende Instruktionen denkbar:

1. Setze einen lokalen Pfad,
2. ändere die Codetabelle,
3. lese Knotenidentifizierungen oder
4. lese einen Satz von Knoten usw.

[0024] In **Fig.** 3 ist ein Attributknoten AN dargestellt. Für diese Art Knoten sind spezielle Knotenidentifizierungen NODE-ID reserviert. Es gibt zwei verschiedene Klassen von Attributen:

1. Solche, die vom Benutzer im Schema definiert sind und bei der Instanziierung mit einem Wert belegt werden können, und
2. Standardattribute, die ebenfalls im Schema definiert werden, aber Einschränkungen an die Syntax des Strukturierten Dokuments zur Folge haben, z. B. wie oft ein Element mindestens oder höchstens auftreten darf usw. Die Knotenidentifizierung NODE-ID legt dabei fest, um welche Klasse und um welchen Typ es sich handelt Attribute sind Blätter im Dokument-Baum, d. h. sie dürfen keine weiteren Kindknoten haben, sondern nur einen Wert. Der Typ des Attributs legt dabei den Typ des Wertes fest, der bei der Übertragung erwartet wird. Es könnte beispielsweise eine Zeichenkette oder eine Zahl sein. Diese Information wird ausgenutzt, um festzustellen, wie viele Bits für das Attribut gelesen werden müssen. Ist der Typ z. B. eine vorzeichenlose ganze Zahl mit acht Bits, dann werden acht Bits für dieses Attribut gelesen, bei einem String wird bis zum ersten Null-Oktett gelesen.

[0025] Inhaltsknoten sind ebenso wie Attributknoten Blätter der Baumstruktur eines Strukturierten Dokuments. In XML gibt es viele vordefinierte Datentypen, die als Attribute oder Inhalte genutzt werden können.

[0026] In **Fig.** 4 ist ein XML-Schema XMLS für ein Beispiel einer Menge von Strukturierten Elementen dargestellt. Hierbei werden u. a. Strukturierte Elemente wie "purchase order type", "address", "items" und entsprechende Variable von diesem Typ definiert.

[0027] In **Fig.** 5 ist ein zum Beispiel von **Fig.** 4 gehöriges binäres Format SBF der, Struktur dargestellt. Die Struktur wird im Zustandsautomatenmodell in die in **Fig.** 5 dargestellte Struktur übertragen, wobei die Namen der Knotenbezeichnungen NODE-ID ID: 1,. . .9 nur zur Referenz in der Textversion von **Fig.** 4 dienen und hier nur beliebige gewählte Zahlen darstellen. Hieraus wird deutlich, dass die gesamte Menge bis auf die Blätter nur Strukturierten Elementen enthält, deren Anzahl von Kindknoten sich im allgemeinen unterscheiden. Ein Decoder muss dies Struktur kennen, um den Bitstrom zur Instanziierung richtig interpretieren zu können. Deshalb muss das Schema beispielsweise vom Server an den Client übertragen werden. Dies kann beispielsweise in Textform oder aber als Anweisung für ein Modell eines Zustandsautomaten erfolgen. Im ersteren Fall muss der Client selbst aus dem Beschreibungsschema in der Textform ein entsprechendes Binärformat des Schemas erzeugen.

[0028] In **Fig.** 6 sind Anweisungen an ein Modell für einen Zustandsautomaten in Form eines Bitcodes BC dargestellt, wobei diese Anweisungen zum Aufbau der Struktur von **Fig.** 4 bzw. 5 führen. Hieraus wird deutlich, dass in einem Befehlsknoten des Zustandsautomaten ein neuer Baumverzweigungsknoten erzeugt wird und zur Erzeugung der Baumstruktur die Zeiger auf die jeweiligen Kindknoten festgelegt werden.

[0029] In **Fig.** 7 ist ein Beispiel eines Strukturierten Dokuments in Form eines XML-Dokuments XML1 dargestellt. Das hier dargestellte Beispieldokument wurde mit dem eingangs definierten Schema aufgebaut. Aus dem hier dargestellten Dokument wird u. a. deutlich, dass den eingangs definierten Strukturelementen hierbei konkrete Inhalte I1, I2, . . . zugeordnet werden.

[0030] In **Fig.** 8 sind Anweisungen BC1 für das durch die baumartige Struktur definierte Modell eines Zustandsautomaten in Form eines Bitcodes zur Erzeugung des in **Fig.** 7 beschriebenen Strukturierten Dokuments dargestellt. Hierbei wird deutlich, dass zur Belegung der Inhaltsknoten nur eine sehr geringe Adressbreite AB benötigt wird, da der Rest der Adressierung quasi durch den jeweiligen Zustand des Zustandsautomaten geliefert wird. Dies hat eine starke Kompression zur Folge, da pro Inhalt nicht immer die volle absolute Adresse für den jeweiligen Inhalt übertragen bzw. gespeichert werden muss. Diesem Beispiel wird der Inhalt zusammen mit dem Baumverzweigungscode übertragen. Der Inhalt könnte aber auch von der Beschreibung abgetrennt in einem eigenen Datenstrom übertragen werden.

[0031] In **Fig.** 9 ist eine binäre Darstellung DBF des Strukturierten Dokuments von **Fig.** 7 dargestellt, wie es durch die Anweisungen von **Fig.** 8 durch den Zustandsautomat, der seinerseits durch die vorher beschriebene Struktur bestimmt ist, erzeugt wird. Das Wurzelelement des Dokuments ist hier in diesem Beispiel "purchase order", seine Knotenidentifizierung NODE-ID ist 20, die hier zufällig für das Beispiel gewählt wurde. Das erste Feld enthält die Zahl der Kindknoten, wobei die Knotenzahl hier 1 ist, da der Vaterknoten und der Befehlsknoten immer vorhanden sind und implizit mitzählen. Die Referenz zum Vaterknoten dieses Knotens ist der Initialisierungsknoten, der alle Dokumente des Terminals verwaltet. Der erste Kindknoten hat den Baumverzweigungscode TBC 01 und die Referenz zeigt auf "purchase order type". Es gibt keinen zweiten Kindknoten, deshalb ist die Referenz 10 NULL. Der letzte Baumverzweigungscode TBC zeigt auf den Befehlsknoten, der aufgerufen werden könnte, wenn dieser Knoten geändert werden soll. Der zweite Knoten "purchase order type" enthält alle Unterelemente, die die detaillierten Informationen zu "purchase order" enthalten, nämlich "ship to", "built to", "comment", "items", "date". Der erste Kindknoten zeigt auf den Vaterknoten, nämlich auf den Knoten mit der Identifizierung NODE-ID 20, also auf "purchase order". Auf diese Weise wird der Dokumentenbaum aufgebaut. Die Inhalts- und Attributsknoten haben keine Baumverzweigungsknoten, sondern nur eine Referenz auf den Vaterknoten. Selbst die Referenz auf den Vaterknoten könnte entfallen, da sie definitionsgemäß die Blätter des Baumes sind und ihrerseits keine Kinder haben können, sondern nur Attributdaten oder Inhalt darstellen. Der Inhalt könnte codiert werden als "inline-Daten", die z. B. durch eine mit Null abgeschlossene Zeichenkette gebildet werden, oder als Zeiger auf einen anderen Datenstrom repräsentiert werden, der separat komprimiert wird. In entsprechender Weise können auch dynamische Veränderungen des Dokuments vorgenommen werden. Im folgenden wird in einem Beispiel gezeigt, wie die Instanz des bestehenden Dokuments dynamisch vom Server mit Hilfe von Anweisungen an ein Modell eines Zustandsautomaten auf der Empfängerseite, also beim Client durchgeführt wird. Angenommen, es soll folgender zusätzlicher Knoten zur Liste der Waren hinzugefügt werden:

```
<item partNum = "724-CN">
<productNamc>Mountain Bikc</productNamc>
<quantity>1</quantity>
<price>3800.00<//price>
<comment>I want it in colour black!</comment>
</item>
```

[0032] Ausgehend von einem in **Fig.** 9 dargestellten Dokumentenbaum beim Client werden vom Server Anweisungen in Form eines Bitcodes BC3 wie er in **Fig.** 10 dargestellt ist, übertragen. Der hier beschriebene Code bewirkt beim Zustandsautomaten des Client im wesentlichen folgendes:

1. Durchführen einer Setup-Sequenz, wobei sich das System während der Initialisierung in einem Befehlsknoten befindet.
2. Gehe zu der Position im Baum, die verändert werden soll. Hierbei ist auch ein vordefinierter Pfad in einer Baumzeigertabelle anwendbar, für den Fall tief verschachtelter Strukturen, auf die oft zugegriffen wird.
3. Gehe zum Befehlsknoten, wobei auch ein rekursiver Aufruf möglich ist.
4. Erweiterung des Baumverzweigungscodes von 2 auf 3 Bits und die Erhöhung der Knotenanzahl von 2 auf 3, wobei sich im Knoten mit der NODE-ID 25 der Inhalt von 11 auf 111 ändert.
5. Danach folgen ähnliche Schritte wie bereits weiter oben erwähnt. Danach ist das System in dem Befehlsknoten und anschließend ist der Zustandsautomat im veränderten Knoten <items>.
6. Um vom Vaterknoten von <items> zurückzukehren, müssen 3 Bits übertragen werden, da die Zahl der Kinder sich von 2 auf 3 erhöht hat.
7. Zum Schluss erfolgt eine Rückkehr zur Wurzel, das wieder eine Instanz des Befehlsknotens darstellt. Das System ist bereit für weitere Befehle.

[0033] Auf dieser Basis ist eine Datenkompression von einer Menge von Strukturierten Elementen bzw. eines Beschreibungsschema, von Strukturierten Dokumenten mit instanziierten Strukturierten Elementen und von entsprechenden Änderungen einer Menge von Strukturierten Elementen oder eines Strukturierten Dokuments möglich, wobei jeweils ein Strom von Anweisungen an einen Zustandsautomat erzeugt wird, der dann entweder gespeichert oder übertragen wird.

[0034] In entsprechend umgekehrter Weise ist natürlich auch eine Dekompression in Strukturen, oder Strukturierte Dokumenten oder deren Änderungen aus den entsprechenden Anweisungen für den zugehörigen Zustandsautomaten möglich.

[0035] Auf dieser Basis sind Anordnungen zur Durchführung des Verfahrens in Form von Encodern oder Decodern realisierbar.

Patentansprüche

1. Verfahren zur Datenkompression eines Strukturierten Dokuments,
bei dem das Strukturierte Dokument in Textform (XML1) vorliegt,
bei dem zu Beginn eine aus mindestens einem Strukturierten Element (TSN, ..) bestehende Struktur (SBF) eines Schemas aufgebaut wird,
bei dem auf der Basis der zu Grunde liegenden Struktur des Schemas Anweisungen (BC1) mit im Wesentlichen relativen Adressen (ID: 20. .53) derart ermittelt werden, dass eine durch das Strukturierte Dokument beschriebene Datenstruktur (DBF) entsteht,
bei dem die Anweisungen (BC1) mit den im Wesentlichen relativen Adressen das Strukturierte Dokument in komprimierter Form darstellen.

2. Verfahren nach Anspruch 1, bei dem die Anweisungen (BC1) binär mit wahlweise variabler oder fester

Länge repräsentiert oder codiert werden, und bei dem diese variable oder feste Länge von der Anzahl der möglichen Verknüpfungen (NC) eines Strukturierten Elementes abhängt.

3. Verfahren nach Anspruch 1 oder 2, bei dem die Strukturen mindestens Baumstrukturknoten (TSN) und zusätzlich Befehlsknoten (IN) und/oder Inhaltsknoten und/oder Attributknoten (AN) aufweisen.

4. Verfahren zur Datenkompression eines Schemas,
bei dem ein Schema (XMLS) für Strukturierte Dokumente in Textform vorliegt,
bei dem auf der Basis mindestens eines bekannten Strukturierten Elements (TSN, . .) Anweisungen (BC) mit im Wesentlichen relativen Adressen (ID: 1. .9, 102. .) derart ermittelt werden, dass eine durch das Schema für Strukturierte Dokumente beschriebene Datenstruktur (SBF) entsteht,
bei dem die Anweisungen mit den im Wesentlichen relativen Adressen das Schema für Strukturierte Dokumente in komprimierter Form darstellen.

5. Verfahren nach Anspruch 1 oder 4, bei dem eine wahlweise variable oder feste Länge der relativen Adressen von der Anzahl der möglichen Verknüpfungen eines Strukturierten Elementes abhängt.

6. Verfahren zur Datenkompression eines Strukturierten Dokuments,
bei dem das Strukturierte Dokument in Textform vorliegt,
bei dem zu Beginn auf der Basis mindestens eines Strukturierten Elements und einem Schema für das Strukturierte Dokument in komprimierter Form eine Datenstruktur aufgebaut wird, die durch das Schema des Strukturierten Dokuments in komprimierter Form beschrieben ist, wobei das Schema für das Strukturierte Dokument in komprimierter Form gemäß dem Verfahren nach Anspruch 4 gebildet wurde,
bei dem auf der Basis der zu Grunde liegenden Struktur für das Schema Anweisungen (BC3) mit Wesentlichen relativen Adressen derart ermittelt werden, dass eine durch das Strukturierte Dokument beschriebene Datenstruktur entsteht, und
bei dem die Anweisungen mit den im Wesentlichen relativen Adressen das Strukturierte Dokument in komprimierter Form darstellen.

7. Verfahren nach Anspruch 1, bei dem das strukturierte Dokument verändert wird und die Veränderung durch die relative Adresse des Operanden im instanzierten strukturiertem Dokument und durch die Adresse einer Operation innerhalb eines Befehlsknotens spezifiziert wird.

8. Verfahren nach Anspruch 4, bei dem das Schema verändert wird und die Veränderung durch die relative Adresse des Operanden im Schema, und die Operation durch die Adresse innerhalb eines Befehlsknoten spezifiziert wird.

9. Verfahrens nach einem der vorhergehenden Ansprüche zur Datenkomprimierung eines XML-Dokuments, eines in einer von XML abgeleiteten Sprache verfassten Dokuments oder eines MPEG7-Dokuments.

10. Verfahren nach einem der vorhergehenden Ansprüche, bei dem aus der komprimierten Darstellung die Struktur des Dokuments oder die Struktur des Schemas ganz oder teilweise rekonstruiert wird.

11. Verfahren nach Anspruch 11, bei dem das strukturierte Dokument oder das Schema ganz oder teilweise in eine textuelle Darstellung decodiert wird.

12. Anordnung zur Durchführung des Verfahrens nach Anspruch 1,
bei der Mittel zum Speichern des Strukturierten Dokuments vorhanden sind,
bei der Mittel vorhanden sind, die auf Basis einer aus mindestens einem Strukturierten Element bestehenden Struktur für ein Schema aufbauen, und
bei der Mittel vorhanden sind, die auf der Basis der zu Grunde liegenden Struktur für das Schema Anweisungen (101) mit Wesentlichen relativen Adressen (101) derart ermitteln, dass eine durch das Strukturierte Dokument beschriebene baumartige Datenstruktur entsteht, wobei die Anweisungen mit den im Wesentlichen relativen Adressen das Strukturierte Dokument in komprimierter Form darstellen.

13. Anordnung zur Durchführung des Verfahrens nach Anspruch 4,
bei der Mittel zum Speichern eines Schemas für Strukturierte Dokumente vorhanden ist und
bei der Mittel vorhanden sind, die auf der Basis mindestens eines bekannten Strukturierten Elements Anweisungen mit im Wesentlichen relativen Adressen derart ermitteln, dass eine durch das Schema für Strukturierte Dokumente beschriebene Datenstruktur entsteht, wobei die Anweisungen mit den im Wesentlichen relativen Adressen das Schema für Strukturierte Dokumente in komprimierter Form darstellen.

Hierzu 16 Seite(n) Zeichnungen

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 1

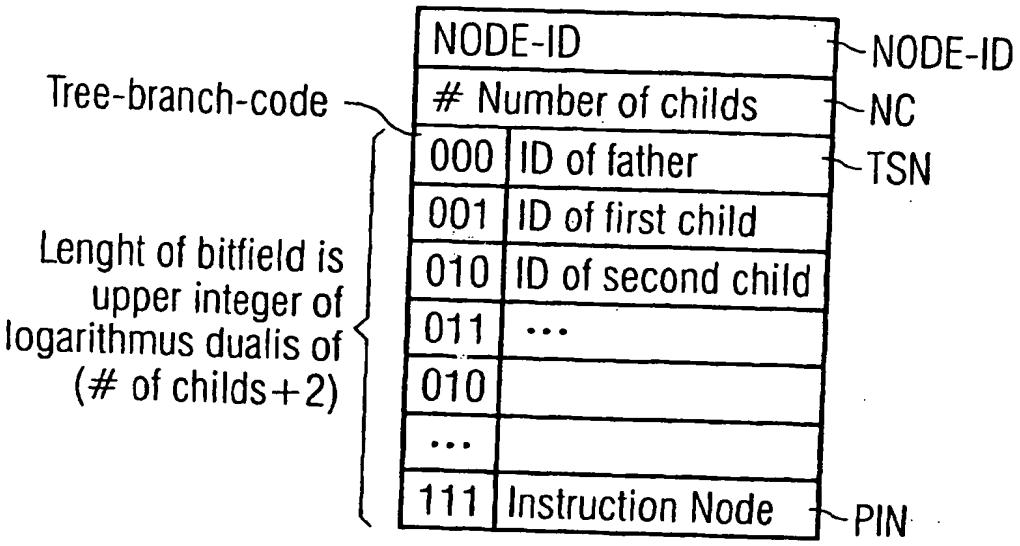
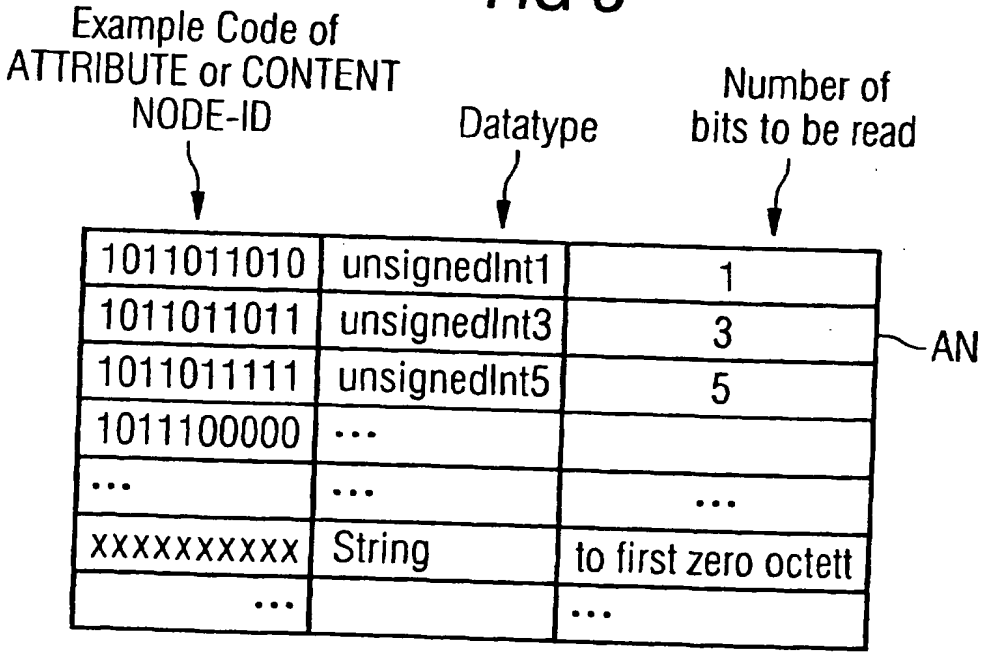


FIG 3



Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

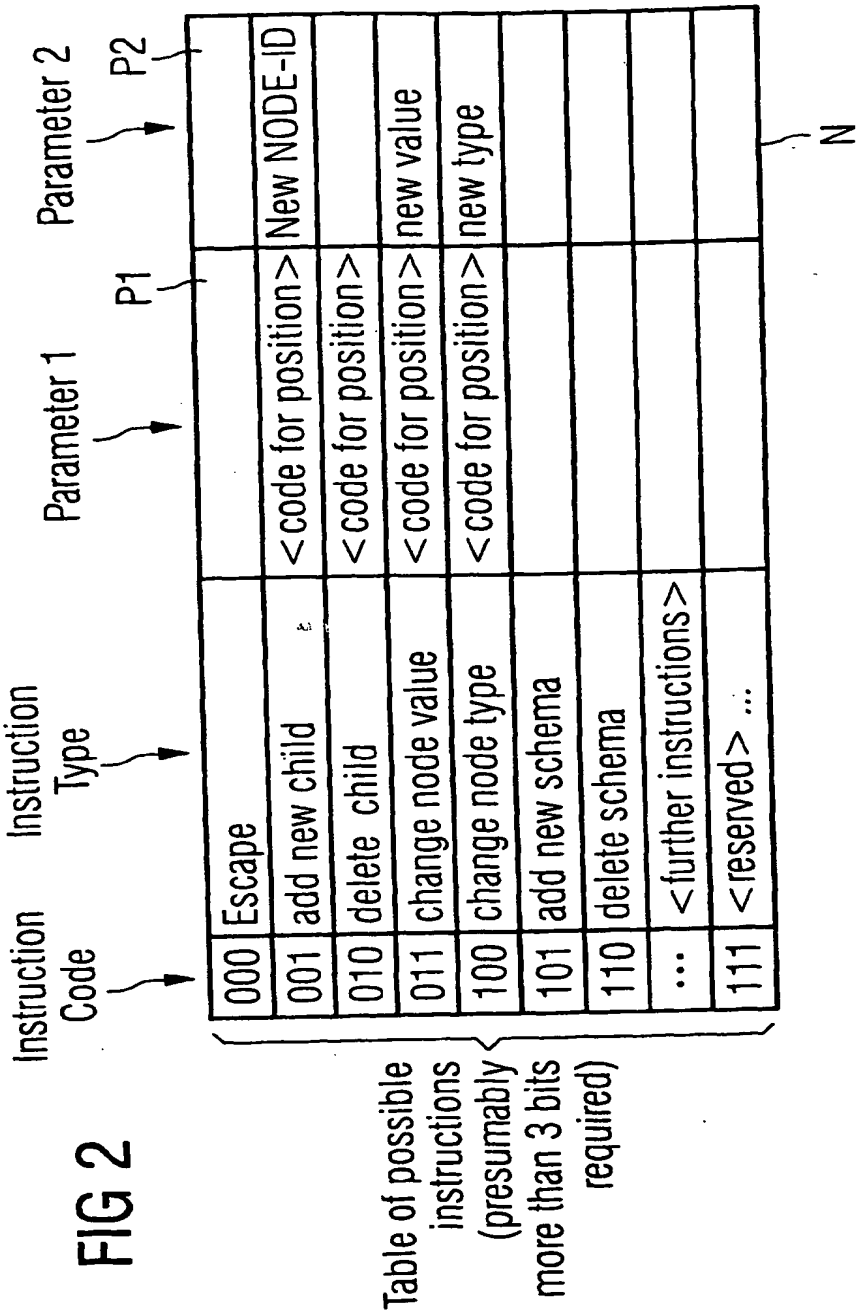


FIG 2

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 4A

XMLS

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <xsd:annotation>
     <xsd:documentation>
       Purchase order schema for Example.com.
       Copyright 2000 Example.com. All rights reserved.
     </xsd:documentation>
    </xsd:annotation>

    <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

    <xsd:element name="comment"    type="xsd:string"/>

    <xsd:complexType name="PurchaseOrderType">
     <xsd:element name="shipTo"       type="Address"/>
     <xsd:element name="billTo"       type="Address"/>
     <xsd:element ref="comment"       minOccurs="0"/>
     <xsd:element name="items"        type="Items"/>
     <xsd:attribute name="orderDate"  type="xsd:date"/>
    </xsd:complexType>

    <xsd:complexType name="Address">
     <xsd:element name="name"         type="xsd:string"/>
     <xsd:element name="street"       type="xsd:string"/>
     <xsd:element name="city"         type="xsd:string"/>
     <xsd:element name="state"        type="xsd:string"/>
     <xsd:element name="zip"          type="xsd:decimal"/>
     <xsd:attribute name="country"    type="xsd:NMTOKEN"
            use="fixed" value="US"/>
    </xsd:complexType
```

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 4B

```
<xsd:complexType name="Items">
 <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
   <xsd:element name="productName" type="xsd:string"/>
   <xsd:element="quantity"
    <xsd:simpleType base="xsd:positiveInteger">
     <xsd:maxExclusive value="100"/>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="price"      type="xsd:decimal"/>
   <xsd:element ref="comment"     minOccurs="0"/>
   <xsd:element name="shipDate" type="xsd:date" minOccurs='0'/>
  <xsd:attribute name="partNum"  type="Sku"/>
  </xsd:complexType>
 </xsd:element>
</xsd:complexType>

</xsd:simpleType name="Sku" base="xsd:string">
  <xsd:pattern value="\d{3}-[A-Z] {2}"/>
 </xsd:simpleType>

</xsd:schema>
```

XMLS

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 5A

SBF

ID:1	<purchaseOrder>
#nodes=1	
00	NULL
01	ID-ref.=2<PurchaseOrderType>
10	NULL
11	<Instruction Node>

ID:2	<PurchaseOrderType>
#nodes=5	
000	NULL
001	ID-ref.=3 <shipTo>
010	ID-ref.=3 <billTo>
011	ID-ref.=8 <comment>
100	ID-ref.=4 <items>
101	ID-ref.=101 <orderDate>
...	<empty>
111	<Instruction Node>

ID:3	<Address>
#nodes=6	
000	NULL
001	ID-ref.=102 <name>
010	ID-ref.=102 <street>
011	ID-ref.=102 <city>
100	ID-ref.=102 <state>
101	ID-ref.=102 <zip>
110	ID-ref.=103 <country>
111	<Instruction Node>

ID:4	<Items>
#nodes=1	
00	NULL
01	ID-ref.=5 <item>
10	NULL
11	<Instruction Node>

ID:5	<item>
#nodes=3	
000	NULL
001	ID-ref.=104<minOccurs>
010	ID-ref.=104<maxOccurs>
011	ID-ref.=6 <anonymous type>
...	<empty>
11	<Instruction Node>

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 5B

ID:6	<anonymous type>
#nodes=6	
000	NULL
001	ID-ref.=102 <productName>
010	ID-ref.=7 <quantity>
011	ID-ref.=105 <price>
100	ID-ref.=8 <comment>
101	ID-ref.=106 <shipDate>
111	ID-ref.=9 <partNum>
...	<empty>
111	<Instruction Node>

ID:7	<quantity>
#nodes=2	
00	NULL
01	ID-ref.=107 <positiveInteger>
10	ID-ref.=108 <maxExclusive>
11	<Instruction Node>

ID:8	<comment>
#nodes=1	
00	NULL
01	ID-ref.=102 <string content>
10	NULL
11	<Instruction Node>

ID:9	<sku>
#nodes=1	
00	NULL
01	ID-ref.=110 <proprietary content>
10	NULL
11	<Instruction Node>

SBF

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

BC

No	Transmitted bitcode	Meaning
1	Xxxxxx	<setup-sequence, tbd.> System is in INSTRUCTION-NODE
2	101	Go to INSTRUCTION "add new schema"
3	01	Parameters for mode of "add new schema"
4	00000001	NODE-ID of schema to be added: <purchaseOrder>
5	00000001	Number of childs is 1
6	01	TBC of first child
7	00000010	NODE-ID of first child
8	00	Back to INSTRUCTION NODE; end of schema with NODE-ID 1
9		
10	101	Go to INSTRUCTION NODE "add new schema"
11	01	Parameters for mode of "add new schema"
12	00000010	NODE-ID of schema to be added: <PurchaseOrderType>
13	00000101	Number of childs is 5
14	001	TBC of first child
15	00000011	NODE-ID of first child is 3
16	010	TBC of second child
17	00000011	NODE-ID of second child is 3
18	<.....>	Rest of <PurchaseOrderType>
19	000	Back to INSTRUCTION NODE; end of schema with NODE-ID 1
	<rest of DS>	

FIG 6A

Nummer: DE 100 47 338 A1
Int. Cl.?: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 6B

Comments to this example:

1	The system is in INSTRUCTION-NODE, bits are interpreted as command. The command appropriate for this situation could be: "define new schema"
2	Code 101 referst to the example-instruction-node of figure 2
3	This parameter defines the mode of the schema-transmission, e.g.: 01 = send TREE-BRANCH-CODE for every child, 10 : do not send TREE-BRANCH-CODE - all childs are transmitted as sequence, terminated by exit-code.
4	The number of bits for a NODE-ID depends on the size of thes code-space, to be defined (in this example 8 bits are assumed)
5	Number of childs is 1. This causes the TREE-BRANCH-CODE to be 2 bits
8	TBC 00 always goes one level higher, in this case to INSTRUCTION NODE

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 7

XML1

The Purchase Order, po.xml

```
<?xml verson="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>          — I1
    <street>123 Maple Street</street> — I2
    <city>Mill Valley</city>            :
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <price>148.95</price>
      <comment>Confirm this is eletric</comment>
    </item
    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <price>39.98</price>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

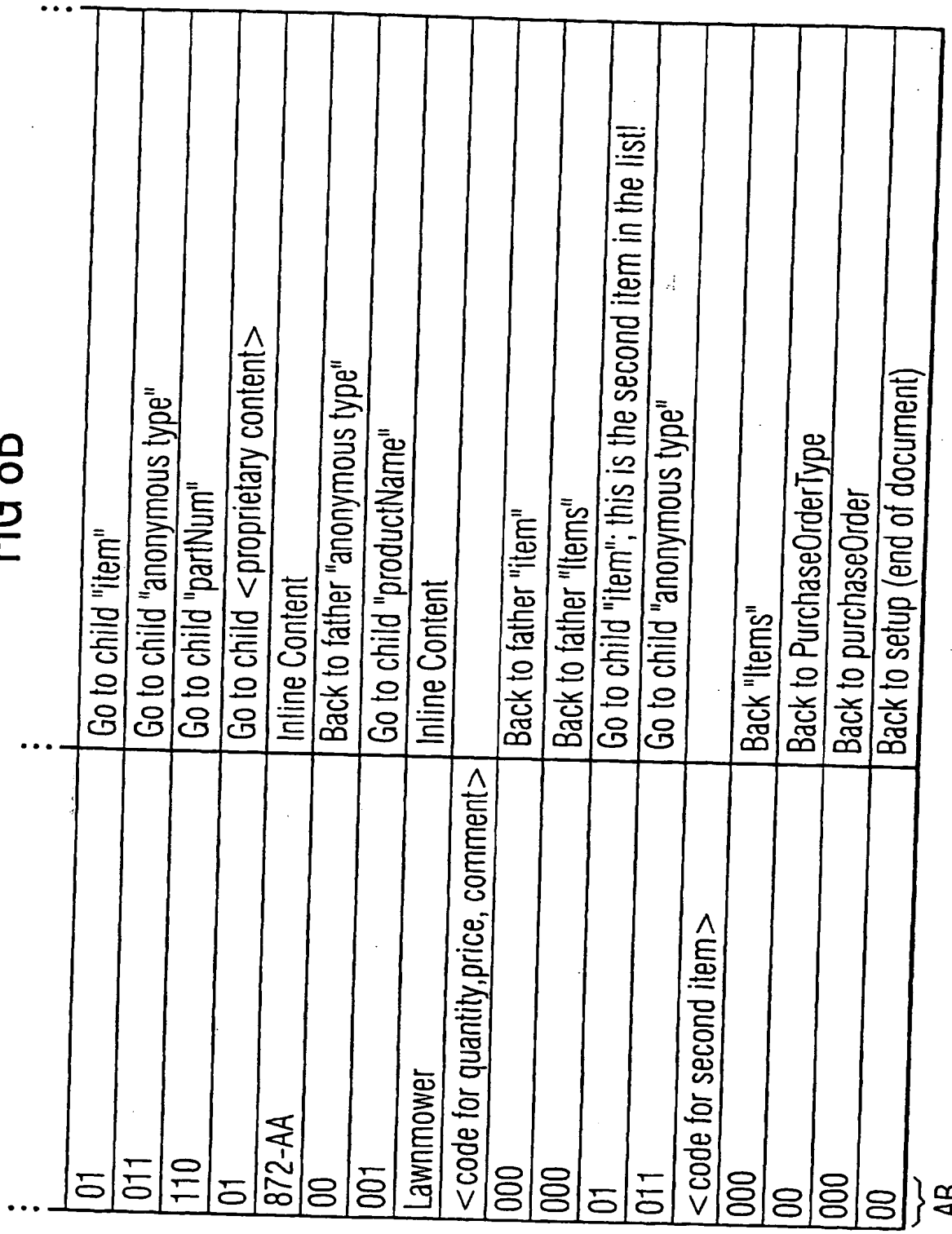
FIG 8A

BC1

Table of binary code, transmitted from the server to the client	
Transmitted bitcode	Meaning
xxxxxxx	<setup-sequence: root element has node-ID 1...> (tbd.)
01	Go to child PurchaseOrderType
101	Go to child orderDate
1999-10-20 (in binary)	Inline content, Attribute node knows how to deal with
001	Go to child "shipTo" in PurchaseOrderType
001	Go to child "name" in address
Alice Smith	Inline content
010	Go to child "street" in address
Maple Street	Inline content
<code for city,state,zip>	
000	Go back to father node of address
010	Go to child "billTo" in PurchaseOrderType
<code for address of billTo>	
000	Go back to fahter node of address
011	Go to node "comment" in PurchaseOrderType
Hurry, my lawn ...	Inline Content
100	Go to child "Items"

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 8B



Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

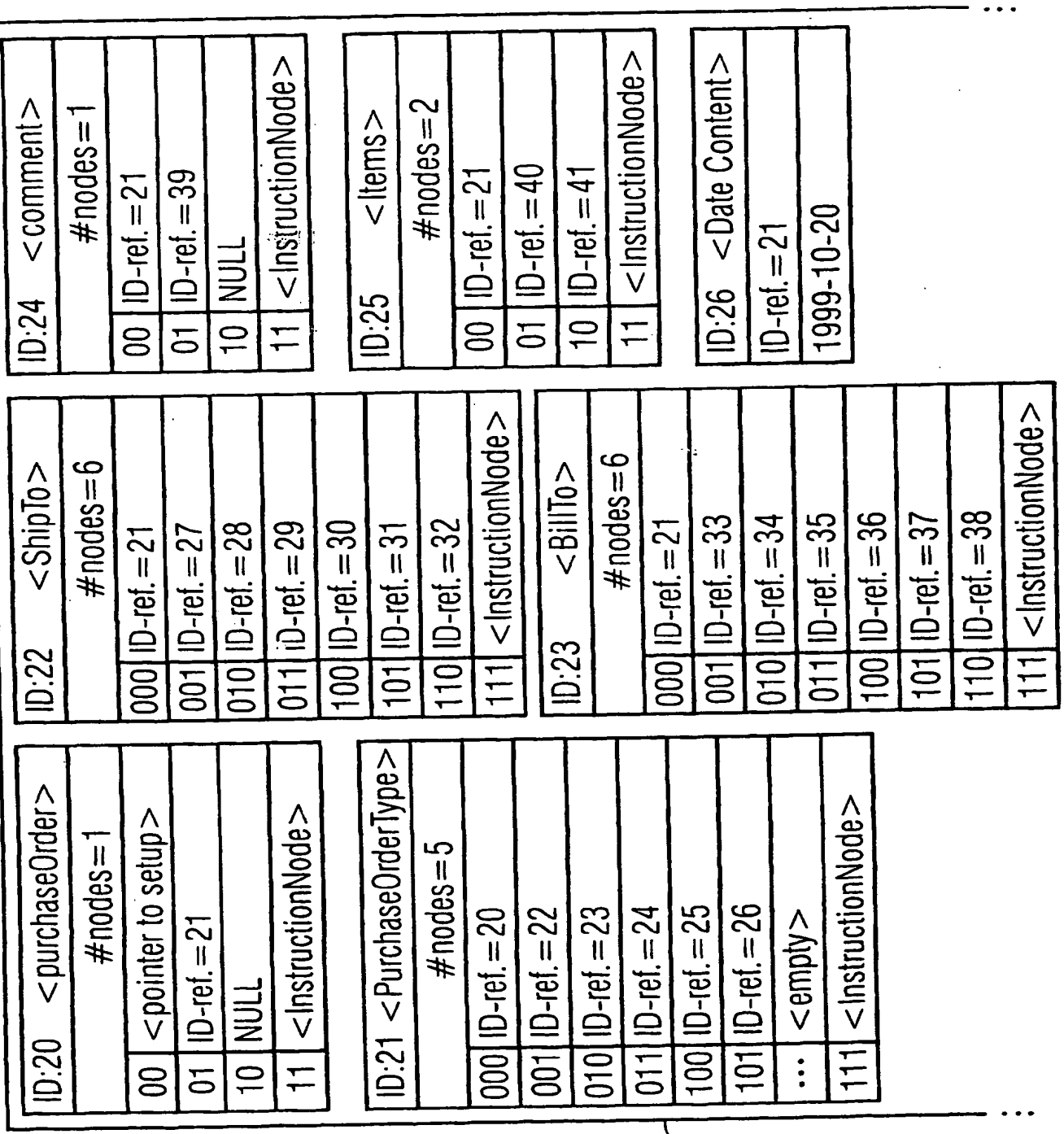


FIG 9A

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

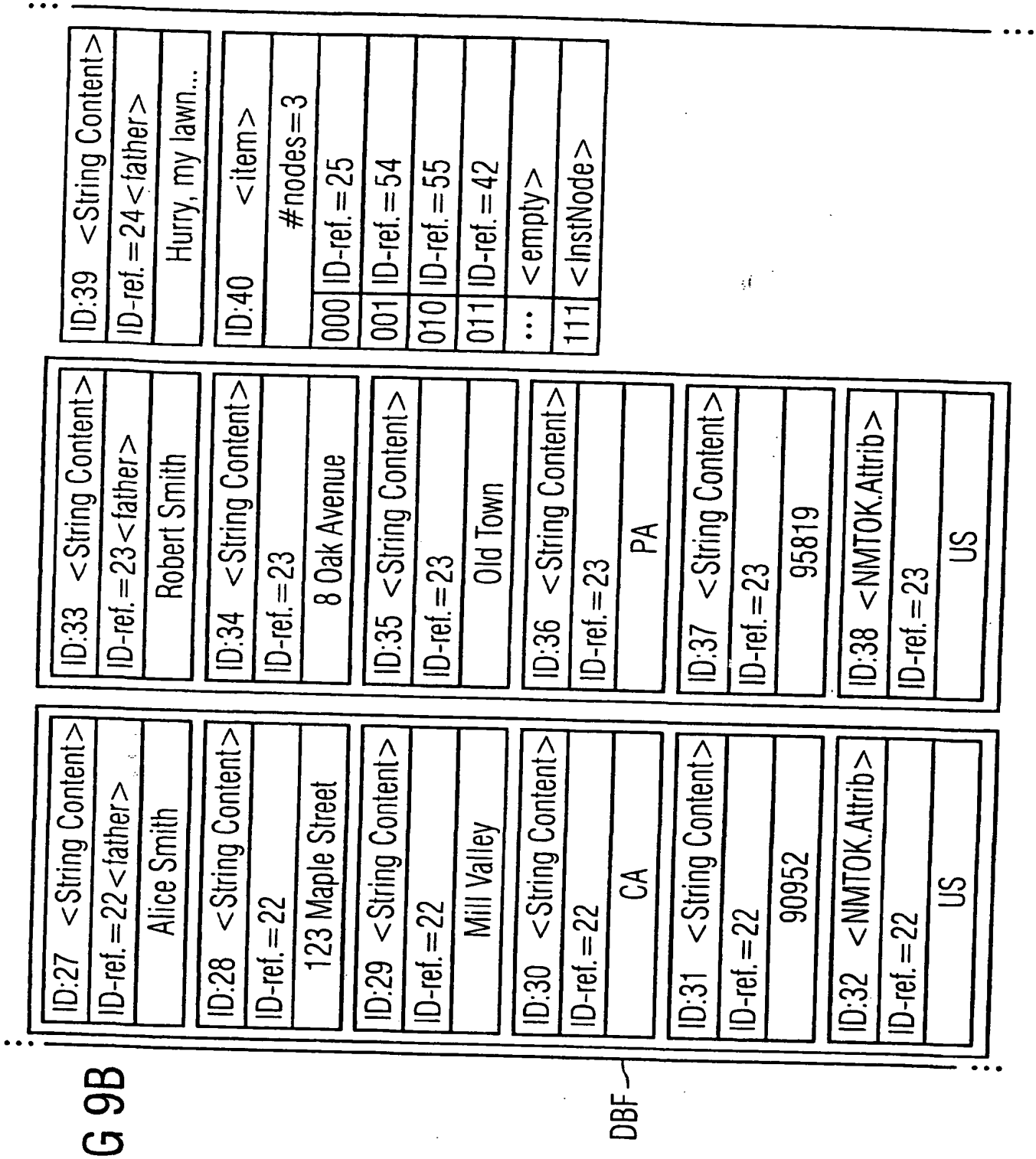


FIG 9B

Nummer: DE 100 47 338 A1
Int. Cl.[7]: G 06 F 17/21
Offenlegungstag: 18. April 2002

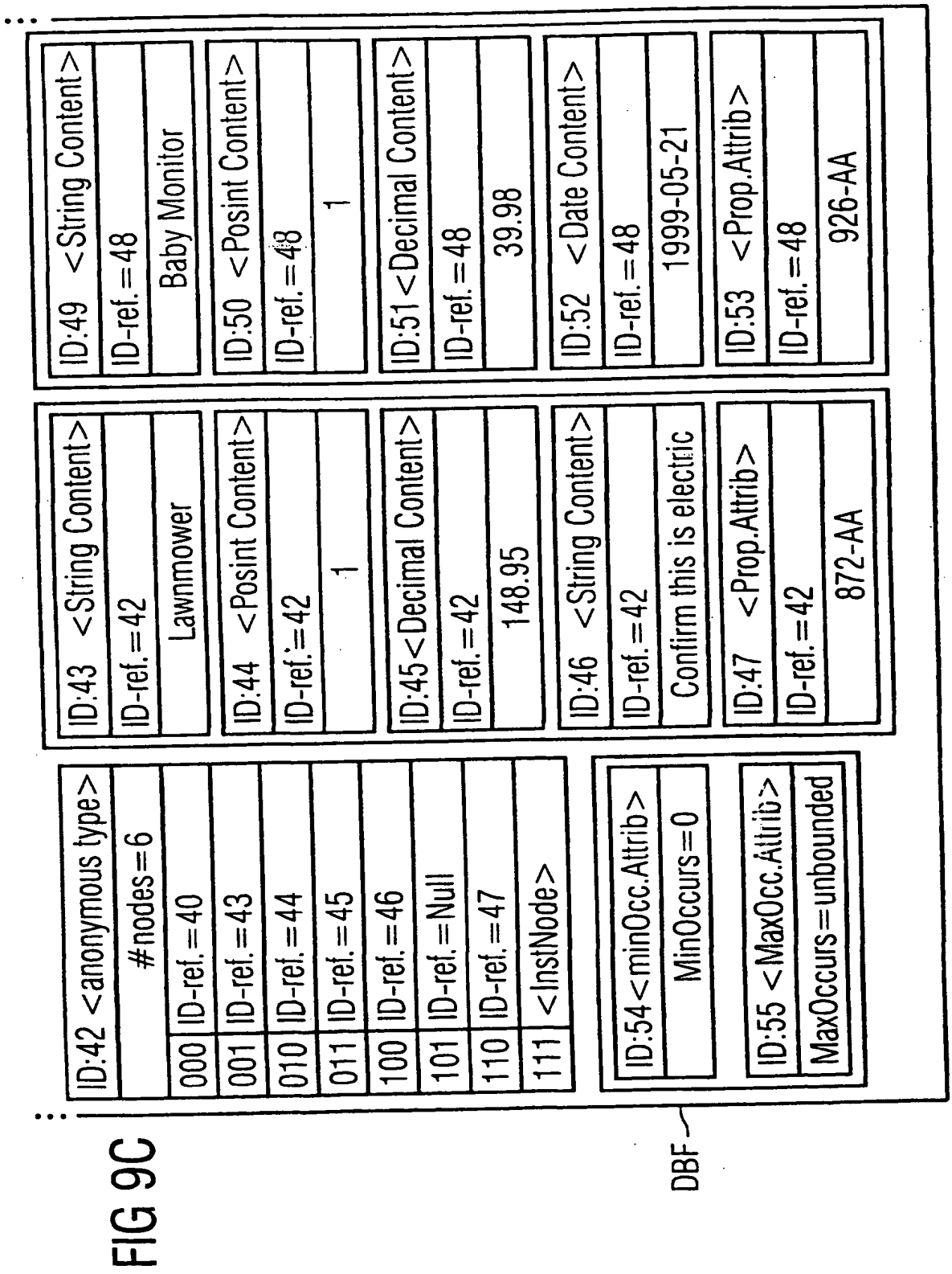


FIG 9C

Nummer: DE 100 47 338 A1
Int. Cl.?: G 06 F 17/21
Offenlegungstag: 18. April 2002

FIG 10A

BC3

No	Transmitted bitcode	Meaning
1	XXXXXX	<setup-sequence: root element has NODE-ID 20>(tbd.)
2	01	Go to NODE-ID 21 <PurchaseOrderType>
3	100	Go to NODE-ID 25 <Items>
4	11	Enter INSTRUCTION NODE
5	001	Instruction is "Add new child"
6	11	Position of child is 3 (first parameter of add-node instruction)
7	00000101	NODE-ID of child is 5:<item>-prototype (lenght of NODE-ID-code here: 8bit)
8	011	Go to child "anonymous type"
9	110	Go to child <proprietary content>
10	724-CN	Inline Content
11	00	Back to father "anonymous type"
12	001	Go to child "productName"
13	Mountain Bike	Inline Content
14	<quantity, price, comment>	
15	000	Back to father "item"
16	000	Back to INSTRUCTION NODE
17	000	Leave INSTRUCTION NODE an go back to NODE-ID:25 <Items>
18	000	Back to father <PurchaseOrderType> *now 3 bits have to be transmitted!*
19	000	Back to father <purchaseOrder>
20	00	Back to <setup>

Nummer:
Int. Cl.?:
Offenlegungstag:
DE 100 47 338 A1
G 06 F 17/21
18. April 2002

FIG10B

Comments to the instructions:

1	Setup-sequence: for initialization, the system is in the INSTRUCTION-NODE. Several instructions are defined.
2-3	Walk the tree to the position, which is to be updated. Also a predefined path in the TREE-POINTER-TABELE is applicable for deeply nested structures, which are often accessed.
4	Enter INSTRUCTION-NODE. It should be possible to enter this node recursively.
6	This requires, to extend the TBC from 2 to 3 bits and #nodes is changed from 2 to 3. The address for the instruction node in the node with NODE-ID 25 changes from 11 to 111.
7	If the NODE-ID of the node to be attached has a different number, than predefined in the schema-definition, then it has to be transmitted. But then the instantiation is not schema-conform anymore!! Presumably it's sufficient to transmit node-type only by TBC of schema <Items>-definition: 01
8-15	This part is equivalent to the initial transmission of the document in table xxx.
16	After this code, the system is in the INSTRUCTION-NODE again, additional instructions could be executed
17	After this code, the system is in the changed <Items>-node.
18	To go to father of Items-node 3bits have to be transmitted, as the number of childs changed from 2 to 3, and 2 bits are not sufficient anymore to code all of them.
19-20	Back to root (this is again an instance of the INSTRUCTION-NODE); The system is ready for further commands.